Vista: Looking Into the Clusters in Very Large Multidimensional Datasets

Keke Chen Ling Liu

College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332
USA
{kekechen, lingliu}@cc.gatech.edu

Abstract

Information Visualization is commonly recognized as a useful method for understanding sophistication in large datasets. In this paper, we introduce an efficient and flexible clustering approach that combines visual clustering and fast disk labelling for very large datasets. This paper has three contributions. First, we propose a framework *Vista* that incorporates information visualization methods into the clustering process in order to enhance the understanding of the intermediate clustering results and allow user to revise the clustering results before disk labelling phase. Second, we introduce a fast and flexible disk-labelling algorithm *ClusterMap*, which utilizes the visual clustering result to improve the overall performance of clustering on very large datasets. Third, we develop a visualization model that maps multidimensional dataset to 2D visualization while preserving or partially preserving clusters. Experiments show that *Vista* combining with *ClusterMap*, is faster and has lower error rate than existing algorithms for very large datasets. It is also flexible because the "cluster map" can be easily adjusted to meet application-specific clustering requirements.

1. INTRODUCTION

Clustering is a common technique used in understanding and manipulating large datasets, such as bioinformatic datasets and high-energy physical datasets. Generally, clustering can be defined as follows: given n data points in a d-dimensional metric space, partition the data points into k clusters such that the data points within a cluster are more similar to each other than data points in different clusters. Clustering algorithms should concern about the following issues: 1) the definition of similarity of data items; 2) the characteristics of clusters, including size, shape and statistical properties; 3) the computation cost and error rate of the result. A key challenge for clustering algorithms on very large datasets is to reduce the time complexity while maintaining low error rate.

Many clustering algorithms have been proposed. Most of them automate the entire clustering process. What user needs to do is setting parameters at the beginning, such as the number of representative points and shrink factor in CURE [3]. Even though most of the algorithms allow people to use different input parameters, once the input arguments are set, the clustering result is produced with no interruption, leaving less flexibility to incorporate any application-specific revision into the clustering rules.

With this problem in mind, we develop a fast and flexible clustering approach for very large datasets. This approach has three unique features. First, it has a framework ***Vista*** that incorporates information visualization methods into clustering process in order to enhance the understanding of the intermediate clustering results

and increase the flexibility of the system. Second, it introduces a fast and flexible disk-labelling algorithm, called ***ClusterMap***, which utilizes the visual clustering result to improve the performance of clustering on very large datasets. Third, it uses a visualization model that maps multidimensional dataset to visualization while preserving clusters partially. This model defines two levels of transformations necessary for visualizing multidimensional datasets. The level one transformations transfer datasets in metric space, non-metric space, or non-Euclidean space to normalized Euclidean space. The level two maps normalized Euclidean space to 2D visual space.

We organize this paper as follows: section 2 presents a closer look at the problems of existing clustering methods; section 3 describes the visualization model; section 4 overviews the visual clustering system *Vista*; section 5 introduces the new labelling algorithm *ClusterMap*; section 6 shows experimental measurement of this approach; finally, we discuss the related work and conclude our work.

2. PROBLEM STATEMENT: A CLOSER LOOK

Performance factors of clustering for very large datasets

As we discussed in introduction, a key challenge for clustering algorithms on very large datasets is to achieve low time complexity and low error rate at the same time. Existing clustering algorithms are generally unsupervised self-learning processes using either *partitional* or *hierarchical* approaches [22] to learn about clusters in datasets. When the datasets become very large, most existing algorithms break down because of the non-linear time complexity (typically, $O(N^2)$ or higher, N is the number of items in dataset) and huge I/O costs. Some new algorithms aim at efficient clustering on very large datasets. The most typical ones are BIRCH and CURE[3,11].

BIRCH and CURE actually follow a framework of clustering for very large datasets, that is sampling → clustering → labelling. No matter what kind of sampling is adopted, (random sampling like in CURE or complicated sampling like CF tree in BIRCH) the goal of sampling is to reduce the problem of clustering on a very large dataset to the problem of clustering on a small representative dataset, then employ lower complexity $O(N)$ labelling on the whole dataset. Both BIRCH and CURE consider the clustering phase to be the dominating factor for accuracy and performance of their clustering algorithms. Thus, they focus on improving the performance of the clustering phase in terms of the time complexity, precision and error rate. However, we observe that the time complexity and error rate obtained from the clustering phase may not be representative of that of the entire three phases. In other word, the time complexity and error rate of the sampling phase and labelling phase should not be disregarded when considering the overall performance of a clustering algorithm. Especially, we found that the labelling phase, if not monitored carefully, can have significant performance impact on the overall clustering process (see section 5).

Factors that affect error rate

Low error rate is the basic goal of clustering algorithms. Existing algorithms make efforts to improve the precision of clustering, such as detecting clusters in arbitrary shapes and eliminating outliers. BIRCH is a very approximate algorithm that can deal precisely with spherical clusters. Its centroid-based labelling is good for statistically distributed clusters similar in size, but it cannot deal with datasets having irregular clusters and lots of outliers. CURE clustering can produce better clustering results on datasets having irregular clusters and lots of outliers, by using representative points to describe convex of a cluster in any shape and size, and by labelling the dataset with representative points.

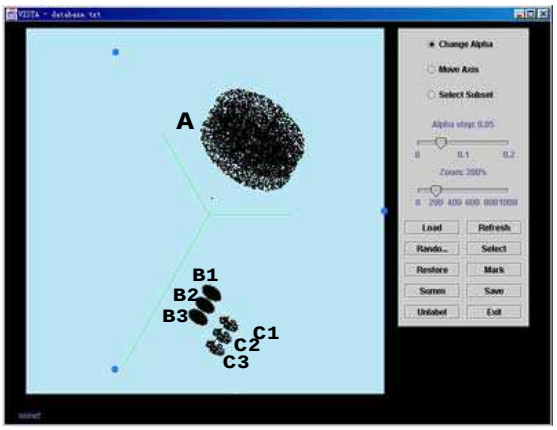

Figure 1. Ambiguity in clustering

However, the CURE algorithm considers all irregular clusters and clusters connected by outliers found in clustering phase as real clusters when labelling the entire dataset. In the situations where irregular clusters or clusters connected by outliers are not real clusters, the CURE algorithm produces clustering result with a much higher error rate. For example, when two clusters connected by outliers should be considered one cluster in application, CURE clustering will introduce errors. Similarly, when an irregular cluster should be divided into one or more small clusters, CURE also introduces errors. Lets take a look at figure 1. This is a 2D projection of a 3-dimensional dataset. It shows there are 7 possible clusters that can be found by CURE. However, C1, C2 and C3 should belong to one cluster according to application-specific semantics actually. Irregular clusters and clusters connected by outliers are common in many scientific data sources, such as those data sources from bioinformatics and high energy physics. Dealing with these clusters correctly is widely recognized as one of the most interesting problems in clustering and data mining. To address this problem, there are two challenges to the existing unsupervised algorithms: 1) How to recognize the application-specific cluster structure? 2) How to flexibly incorporate domain knowledge into the clustering process to reduce the error rate as early as possible?

Cluster visualization

Information visualization is commonly recognized as a useful method for understanding sophistication in large datasets. In this paper we propose a fast and flexible clustering approach for large multidimensional datasets that combines visual clustering with effective disk labelling. We implement this approach in a visual clustering system *Vista*. *Vista* addresses the performance and error rate problems by incorporating information visualization techniques into the clustering process. Formal studies [8] supported the notion that visual exploration can help in cognition. It is especially useful when little is known about the data and the exploration goals are vague. Because user can participate in the exploration process, user can use his/her domain knowledge to adjust goals automatically through the interactive interface of visualization. In addition to get user involved, visual data exploration introduces several main advantages over the automatic data mining techniques in statistics and machine learning: 1) It deals more easily with highly heterogeneous and noisy data; 2) It is intuitive; and 3) It requires no understanding of complex mathematical or statistical algorithms or parameters [13]. As a result, visual data exploration often delivers better results especially in cases where automatic algorithms fail.

However, detecting clusters in visualization brings up three special problems: 1) the limited system capability, e.g. memory and CPU, may restrict the size of the datasets that can be visualized in real time. 2)

How to visualize the clusters without introducing too much visual bias. This is known as the problem of cluster preserving. If the visual bias cannot be avoided, what are the mechanisms we can use to improve the clustering quality? 3) Time cost of human interaction. Although visualization methods tend to deliver better results, human operations are usually slower than automatic algorithms on small datasets.

We propose our solutions to address the three problems of cluster visualization. 1) In *Vista*, We use the sampling Ą clustering Ą labelling three-phase model to address the first problem. More concretely, we use sampling to generate representative dataset that is manageable in size. Then we derive cluster patterns from this sample set, which are then applied to label the entire dataset. 2) Solutions to the second problem depend on the underlying mechanisms used for visualization, such as scatterplot or parallel coordinate. In the first version of *Vista*, we use the scatterplot like visualization because we believe it is the most intuitive way to visualize clusters. However, the advantage of visualization does not come for free. This mechanism may introduce two kinds of visual discrimination: visualization may break a cluster into two clusters or may present misconception that two clusters are seen as one visually. In *Vista*, a visualization model that maps raw dataset onto 2D space is used to avoid breaking clusters, and interactive operations are used to improve cluster quality and discriminate new clusters. 3) The third problem is addressed in two ways. First, we use the mechanisms such as dynamic interaction to improve the clustering quality and the human's understanding of the clusters. Second, we optimise the post-clustering process, namely, introducing efficient encoding of the visual clustering results in "cluster maps" and a fast labelling algorithm, to reduce the overall time complexity.

The rest of the paper proceeds as follows. We first describe a model for creating interactive visualization; then we give an overview of *Vista* and introduce the *ClusterMap* algorithm. We also discuss our experiments on the usability of *Vista* and on the effectiveness and cost of the *ClusterMap* algorithm in the following sections.

3. A MODEL TO CREATE INTERACTIVE VISUALIZATION

Various multidimensional data visualization methods are proposed, while we are interested in scatterplot like visualization. To visualize datasets in a scatterplot, we expect to see clusters are visualized as point-dense regions. It is the most intuitive way for human to find clusters in visualization. Visualizing a dataset inevitably involves a series of transformations. To guarantee the intuitive results are theoretically correct, these transformations should at least approximately preserve the distance relationship between original data points.

The key idea in this model is to transfer any kind of distance into Euclidean distance and then map the Euclidean distance to visible 2D Euclidean distance while preserving distance relationship between points.

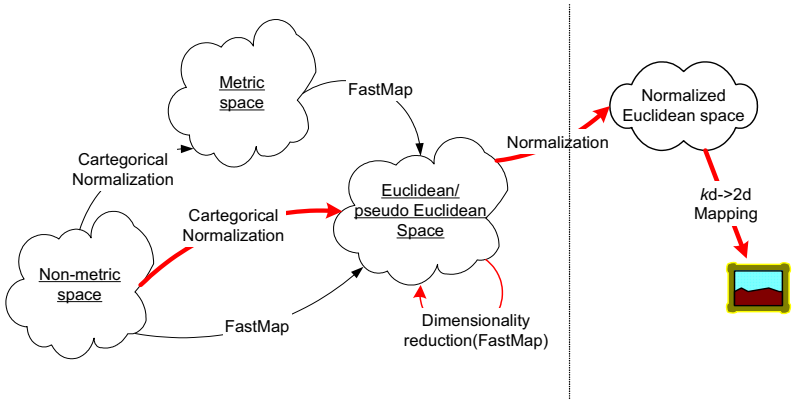

Figure 2: Transformations from multidimensional raw dataset to 2D visual space

However, it is inevitable to introduce visual bias when mapping *k*-d dataset to 2d dataset, such as overlapped clusters. We want to provide a highly interactive visualization that enables user to operate the visualization with adjustable parameter on each dimension to reveal all clusters.

We consider five possible data spaces involved in our framework: non-metric, metric, Euclidean, normalized Euclidean, and 2D visual Euclidean space. Datasets that contains non-numeric data, such as categorical or Boolean data, are in non-metric space. Non-metric space can be normalized to metric space. Many practical datasets are defined in Euclidean space. However, non-Euclidean datasets can also be transformed to Euclidean datasets if distances between points are given. Normalized Euclidean space contains only numeric data located in [0, 1] that is used to create visualization in scalable space further.

Following discussion will focus on: 1) Normalization 2) Transformation of Non-Euclidean space to Euclidean space or dimensionality reduction 3) *k*d to 2d Euclidean Mapping. We discuss the methods used in our *Vista* system mainly. However, any reasonable transformation can be used in each stage. We assume the datasets used for clustering are in form of relational tables, or rows by columns.

3.1 Normalizations in the context of sampling

We discuss two kinds of normalizations: one is used to convert non-numeric data to numeric data; the other is used to scale datasets into [0, 1].

Categorical Normalization

The first one is used to represent non-numeric data in numeric way that facilitate the further numeric based transformations. Many raw datasets include non-numeric columns. Usually this kind of data has no special metric meaning, thus, transforming non-numeric data into numeric data may be an arbitrary process. We use a simple hash function to map a non-numeric item, regarded as a string, to an integer value in [0, *n*], *n* is the number of different strings. With such kind of mapping, if most data columns are non-numeric, visualization results may be greatly affected by different hash functions because this normalization affects the distance relationship greatly. However, in Vista visualization model, this normalization is quite useful to observe the effect of categorical attributes to the whole dataset. We will introduce a flexible way to utilize this normalization in the following part.

Max-Min Normalization

The second kind of normalization is used to scale datasets into [0, 1]. Scaling is used to fit the visualization into certain display area. A bunch of normalization methods are described in [20], among which, min-max normalization is a linear transformation. It does not introduce any potential bias into the dataset and eliminate the side effect of individual column to the entire dataset. Max-min normalization is simple: having maximum and minimum bounds (max and min) of a column, following transformation is used to scale all items into [0, 1]:

A collection of normalization methods are described in [16], among which, min-max normalization is a linear transformation. The goal of using max-min normalization in *Vista* visualization model is to prepare the data points for the succeeded B mapping. The max-min normalization is simple: Given the maximum and minimum bounds (max and min) of a column, it defines the following transformation to scale all items in this column into [0, 1]:

$$\textit{max-min normalization: } v' = (v - \min)/(\max - \min),$$

v is the original value and *v'* is the normalized value.

One problem with the max-min normalization is the possibility of encountering "out of bounds" error, when the three-phase clustering model is employed for large datasets. Concretely, at the clustering phase, the dataset that participates in normalization is a sample set of the raw dataset. Therefore, the max and min bounds used for the sample dataset may not be the bounds for the entire raw dataset. To handle the "out of bounds" error, we propose a modified max-min algorithm to minimize the impact of "out of bounds" while still taking the advantage of original max-min normalization. In this modified algorithm, two questions need to be addressed: (1) If the "out of bounds" data appears, how to handle them; and (2) how to minimize the probability of data items "out of bounds". One feasible assumption is that the distribution of each column in a very large dataset can be approximately modelled as normal distribution, i.e., given mean σ and variance ω^2, the distribution can be determined [20]. Because the data to be normalized is randomly sampled from the raw dataset and the sample size is large enough to preserve the properties of the raw dataset, according to the principles in statistics [20], the sample dataset can approximately describe the statistical properties of the original one. Let $\bar{x}$ denote the average value of column *i*, *n* is the number of rows in the sample dataset, and x_j is the value of the item in row *j* of column *i*. The mean and variance of a column can be approximated by:

$$\sigma_i - \bar{x} \mid \frac{1}{n}\sum_{j|1}^{n} x_j \qquad \omega^2{}_i - \frac{1}{n 4 1}\sum_{j|1}^{n}(x_j 4 \bar{x})^2$$

Let the probability of data items in the raw dataset that are out of bounds be limited to κ and b_i denote the distance from bounds to the mean σ_i. With *Chebyshev's Inequality* [20], the distance from bounds to the mean σ_i can be computed as follows:

$$P\{|X-\sigma_i| \emptyset b_i\} \Omega \omega_i^2 \big/ b_i^2 = \kappa \quad \text{♥} \quad b_i = \sqrt{\omega^2{}_i \big/ \kappa}$$

So the bounds of a column can be approximately defined as:

$$\left(\sigma_i 4 \sqrt{\omega^2{}_i \big/ \kappa}, \sigma_i 2 \sqrt{\omega^2{}_i \big/ \kappa} \right\rfloor$$

We want the visualization of the sample set as precise as possible, which requires the errors introduced in the preprocessing stage to be minimized. With the above bounds, it is still possible to introduce "out of bounds" errors into the sample set. To ensure the sample data will not be out of bounds, we need to modify the above bounds a little bit. Let min(sc_i) and max(sc_i) are minimum and maximum values of column *i* in the sample data. The bounds of each column sc_i are adjusted to:

$$\left(\min\{\min(\ sc_i), \sigma_i 4 \sqrt{\omega^2{}_i \big/ \kappa}\} \ , \ \max\{\max(\ sc_i), \sigma_i 2 \sqrt{\omega^2{}_i \big/ \kappa}\} \right\rfloor$$

In the first prototype of *Vista*, we set κ = ¼, which relaxes the boundary to 2 times of the central area, and yield the bounds:

$$\left\lfloor \min\{\min(\ sc_i), \sigma_i 4 2\omega\} \ , \ \max\{\max(\ sc_i), \sigma_i 2 2\omega\} \right\rfloor .$$

All of the parameters used to normalize all of the columns can be calculated by scanning the sample set twice. When the normalization is applied to the raw dataset later on, a value in column *i*, if out of bounds, can be scaled to the nearby bound.

3.2 Transform non-Euclidean Metric space to Euclidean space or to Reduce Dimensionality

Since human is visually sensitive to 2D Euclidean distance, we expect user can simply justify the clusters on screen with density of point clouds. Any non-Euclidean data needs to be transformed to Euclidean data first. The distance relationship should be preserved approximately in this process. Any non-Euclidean data space

can be mapped into Euclidean or pseudo Euclidean data space where distance can be negative, if the distance function is defined. Let x, y be any two points in a space, function d(x,y) is called distance function if it satisfies:

1) d(x, x)=0, and d(x, y)>0 if x!=y --- non-negative
2) d(x, y)=d(y, x) --- symmetry
3) d(x, y)<=d(x, z)+d(y, z) --- triangle inequality.

Usually, the similarity or dissimilarity function of two data items can be represented by the distance of the two items when clustering algorithms are applied. Thus, for any kind of clustering task, distance function or distances are always given. Given distances between all items, a transformation aiming at distance preserving, like *FastMap*, [14] can transform non-Euclidean space to Euclidean space and preserve the distances approximately. *FastMap* can also be used to reduce dimensionality of data space and thus, enable to produce an operable and understandable visualization. Any other transformations [15] that preserve distance can be applied in this phase.

There is one main disadvantage of this transformation applied in Vista cluster rendering system. After this transformation, the new dimensions have no actual meaning like those defined in the original dataset. Thus, the user loses the opportunity to explore the dataset with more semantics and operations we discussed later.

3.3 *B* Mapping - *k*d to 2d mapping in Eucliean space

Because human vision is only able to percept 2D or 3D Euclidean space, some kind of mapping is needed to transform multidimensional Euclidean data space to 2D/3D visual space. Visualizing multidimensional dataset is a problem for long time. *B* mapping, initially proposed in [1], is a linear *k*d to 2d mapping and it also enables to create some kind of interactive visualization.

To make the paper self-contained, we briefly describe the definition of *B* mapping here. Let R^k be a *k*-dimensional (*k*-d) data space and R^2 be the two-dimensional (2-d) space. Let $x_i(p)$ denote the value of the *i*-th coordinate of a point $p(x_1, x_2, ..., x_k)$ in the *k*-d data space, which has been normalized to [0, 1]. Let q(x, y) denote the point in R^2 space. *B* maps each data item $p(x_1, x_2, ..., x_k)$ in R^k to a point q(x,y) in R^2 as following: Let C be a circle with center c and radius *r* in R^2 (The value *r* depends on the area of the display surface) (see figure 3) . Let $s_1, s_2, ..., s_k$ be *k* equidistant points on the circumference of C. Let $\overrightarrow{cs_i}$ represent the dimension *i* of R^k in R^2, and $\overrightarrow{cq}$ denote the vector sum of $\overrightarrow{cs_i}$ with weight $\zeta_i * (2/k) * (x_i(p) - 0.5)$, in which the ζ_i (*i* = 1,2,…*k*) are adjustable parameters corresponding to each dimension, and $-1 \le \zeta_i \le 1$.

$$\overrightarrow{cq} = \sum_{1 \le i \le k} \zeta_i (2/k)(x_i(p) - 0.5) \overrightarrow{cs_i} \qquad (1)$$

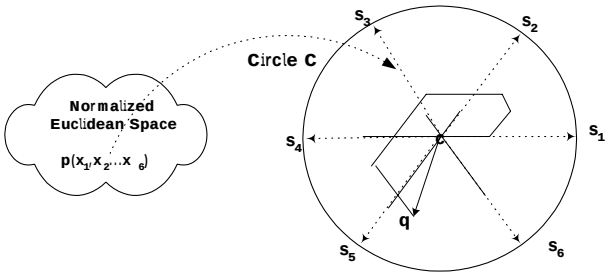

Figure 3. Illustration of *B* mapping with k=6

The $x_i(p)$ of $p(x_1, x_2, ..., x_k)$ is required to be normalized to [0, 1] before mapping. This scaling is used to fit the visualization into certain display area. Figure 3 shows an example B mapping with *k=6.* Each s_i (i=1,…,6) on the circle C represents a column of the data items in the 6-dimensional dataset.

To prove the visualizations produced by the *Vista* visualization model is reliable, we need to prove that *Vista* visualization model is a linear mapping model. This can be proceeded in two steps: First, we prove the mapping function B is a linear mapping, given the constants ζ_i , i =1, 2, …k.[19]. Then we use the basic principle of linear mapping to show that the composition of linear mappings is still linear mapping. It is known that the linear mapping model does not break clusters but may cause overlapped clusters, and sometimes, clustered outliers [19].

Property1:

Mapping function B is a linear mapping that maps k-d data space to 2-d data space, given constants ζ_i , i =1, 2, …k.

[Sketch of Proof] Let us take a look at the mapping function (1) again. It can be represented as

$$\overrightarrow{cq} = \sum_{i=1}^{k} \zeta_i (2/k)(x_i(p) - 0.5)\ \overrightarrow{cs_i}$$

$$= \frac{2}{k}\left[\overrightarrow{cs_1}, \overrightarrow{cs_2}, ..., \overrightarrow{cs_k}\right] \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_k \end{pmatrix} - \frac{2}{k}\left[\overrightarrow{cs_1}, \overrightarrow{cs_2}, ..., \overrightarrow{cs_k}\right] \begin{pmatrix} 0.5 \\ 0.5 \\ ... \\ 0.5 \end{pmatrix}$$

$$= A\overrightarrow{X} - B$$

where,

$$A = \frac{2}{k}\left[\overrightarrow{cs_1}, \overrightarrow{cs_2}, ..., \overrightarrow{cs_k}\right], \quad B = \frac{2}{k}\left[\overrightarrow{cs_1}, \overrightarrow{cs_2}, ..., \overrightarrow{cs_k}\right] \begin{pmatrix} 0.5 \\ 0.5 \\ ... \\ 0.5 \end{pmatrix}, \text{ and } \overrightarrow{X} = (x_1, x_2, ..., x_k)$$

For a certain group of ζ_i, *A* and *B* are constant matrix, so mapping *B* is a linear transformation that actually projects a *k*-d dataset into a 2-d plane determined by ζ_i*, i =1, 2, …k.*

Property2:

The mapping, max-min normalization ∫ B mapping, is a linear mapping.

[Sketch of Proof]

From Property 1 and the previous discussion, we have known that both max-min normalization and B mapping are linear mapping. The composition of linear mappings is a linear mapping [21]

Property3:
Linear mapping will not "break" clusters, but may cause clusters to be overlapped.
[Sketch of Proof]
With linear continuous transformation (actually projection to 2-d plane), two points that are close in R^k will remain close in R^2 [9], which means linear mapping does not break clusters. However, two points that are close in a 2-d projection need not be close in R^k, which means clusters, after linear mapping, may overlap with each other and the outliers may be falsely clustered.

We have mentioned the normalization to categorical data columns. Categorical data columns in a dataset probably destroy the distance preserving property of B mapping. However, our Vista cluster rendering system is not limited in cluster finding. It can also be used to explore the effect of one or some attributes to the whole dataset, for example, the clusters formed by one or several categorical attributes. We illustrate how to observe the clusters formed by exploring one categorical attribute. We suppose the categorical attribute is s_2 in Figure 3 and it has 3 categories, among which category B is the largest one with the most items in it, A is the middle one and C is smaller than the other two (as in Figure 4). Adjusting the ζ value of the categorical attribute, a user can usually observe a 3-cluster visualization dynamically – even the data in one cluster may be sparse but they tend to move together and have different moving behaviours (speed and direction) from those in other clusters. The three clusters – A, B and C, are equidistantly distributed along the s_2 direction if the side effect of the reverse dimension s_5 is reduced – setting the ζ of s_5 to a small value.

To help users observe and separate the overlapped clusters and the possibly clustered outliers, we have developed several visual interaction techniques for the *Vista* cluster rendering system. In the next section we give a brief overview of the *Vista* visual clustering system and demonstrate how to apply the interaction techniques to discover the potentially overlapped clusters and outliers. For a more detailed discussion, see our technical report [19].

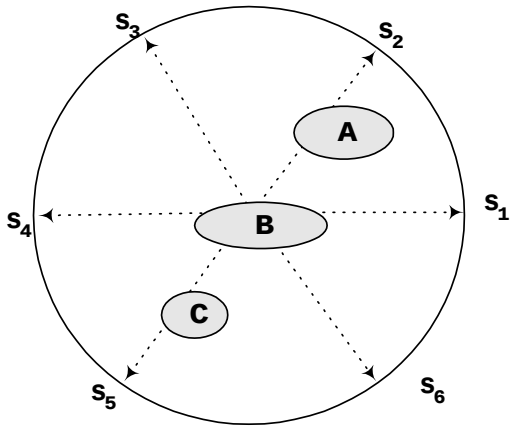

Figure 4. The clusters formed by a categorical dimension

4. *VISTA* – AN INTERACTIVE VISUAL CLUSTERING SYSTEM

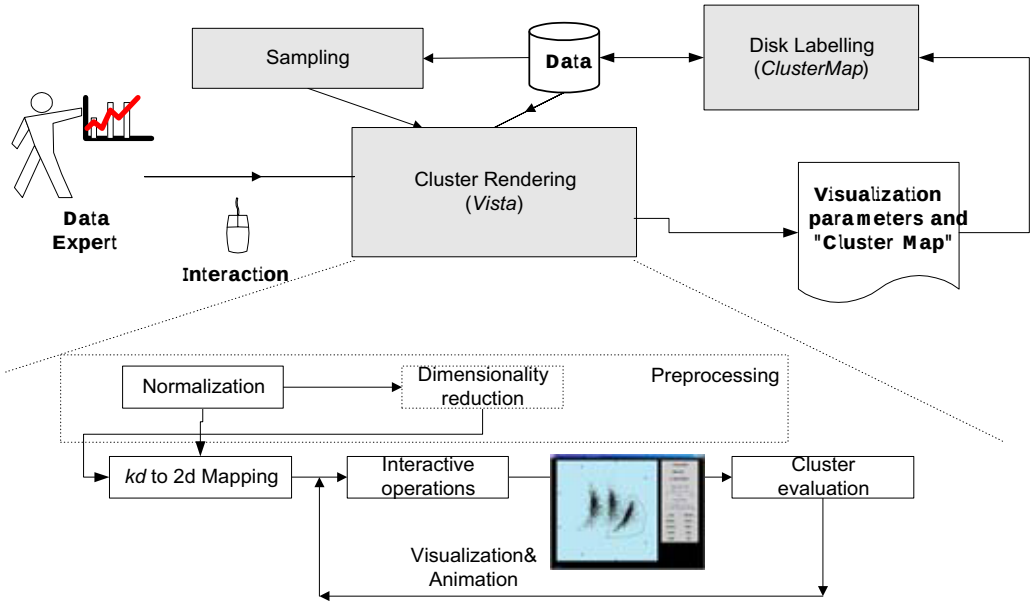

Figure 5: *Vista* system framework

4.1 The system framework

Based on the mapping model, we build an interactive visual clustering system - *Vista*. Our system is designed for Euclidean datasets in current version, so *FastMap* is used to reduce dimensionality only. This system employs the 3-phase clustering model for very large multidimensional datasets. It can also be used to import clustering results produced by existing algorithms to understand the relationship between clusters and then label the whole large dataset in a fast and reliable way.

Sampling: For very large dataset, random sampling is employed to reduce the dataset to subset in an appropriate size. An appropriate size means not only the subset can be processed in limited memory and the system can create smooth visualization in real time, but also the subset is good representative that preserves structure of whole dataset approximately. Guha[3] gave an estimate that, in order not to miss the smallest cluster in size of x, how large a sample size should be, given the size of dataset and the minimum probability of missing a cluster.

Normalization: We assume the raw dataset in structure of rows and columns. The raw dataset may include non-numeric columns, like categorical or boolean data. For further distance calculation, non-numeric data items are normalized to numeric data. And then all data items are scaled to range [0, 1] to fit the scalable display area.

FastMap: An understandable visualization accommodate only limited number of dimensions because of the limitation of human visual ability. Too much visual information usually causes the difficulty of understanding. Although our system introduce a new dimension visualization method that enables user to

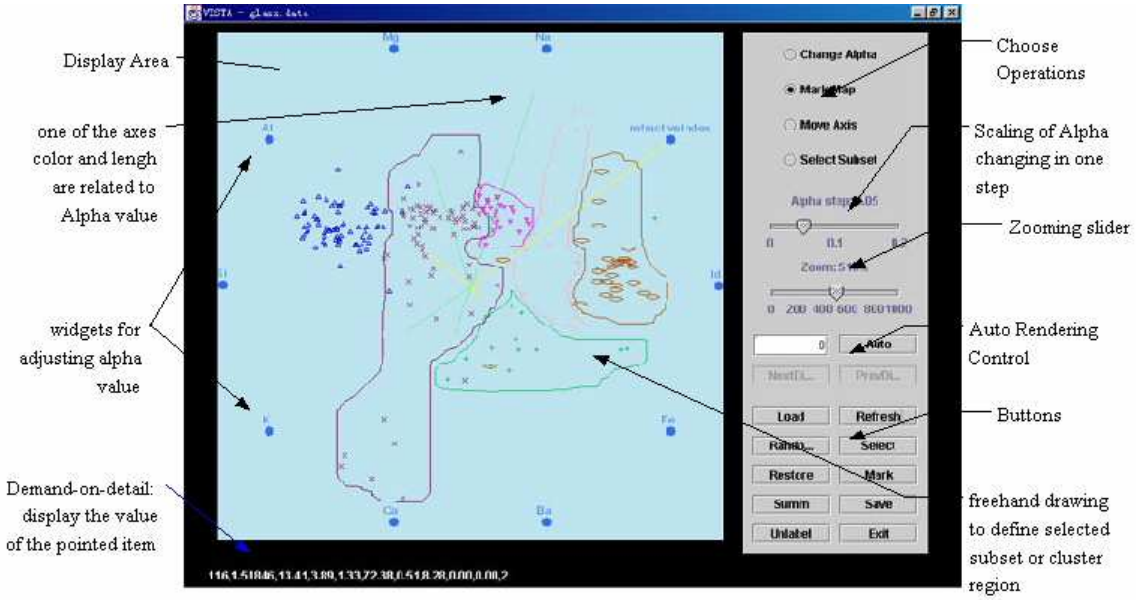

Figure 6: User interface of *Vista* system

operate with up to 40~50 dimensions, datasets having higher dimensions would impose too much burden on user's perception. *FastMap* is used to reduce the dimensionality to an acceptable number.

***k*d->2d *B*′ Mapping**: Because the visualization space in *Vista* system is 2D, high-dimensional (>2D) dataset should be mapped into 2D visual space. Linear mapping is desired to at least preserve clusters partially.

Interactive Operations: interactive operations enable user to operate the visualization directly and, thus, visually understand and find the ideal cluster distribution. These operations include ζ parameter adjustment, random rendering, subset selection, zooming, cluster marking&unmarking, and labelling. Continuous interactive operations, mainly parameter adjustment, can create animation, with which user can get more information than with static visualization. When a satisfactory visualization is found, user can define cluster region by freehand drawing the boundaries (Figure5). The defined cluster regions are saved as "cluster map".

Cluster Evaluation: Cluster size, cluster density, distances between clusters and validity index S_Dbw [7] are calculated when a cluster view is defined. They help user evaluate the quality of possible clusters. Validity indices are kept in each step and the visualizations can be retraced by validity indices like in Projection Pursuit [17].

Parameter Saving: Parameters creating visualization are saved for restoring visualization next time. Parameters and "cluster map" are also used to perform labelling algorithm *ClusterMap.*

Disk labelling: For large dataset, this step is going to label the entire dataset. We implement a region based labelling algorithm- *ClusterMap*.

4.2 GUI and Interactive Operations

The goal of interactive operations is to observe the dataset from different projection planes and help users to recognize those overlapped clusters or falsely clustered outliers. A set of interactive operations is provided in the *Vista* cluster rendering system.

- ζ **parameter adjustment** changes the projection plane and redisplay the dataset in real time.

- **Subset selection by region** defines a subset by freehand drawing an enclosed region on screen, which can be used for further processing, such as cluster marking or showing only the selected subset.
- **Subset selection by range selection** defines a subset by selecting a range of one dimension.
- **Zooming** helps users explore the details or see an overview of the dataset.
- **Focus changing** shifts the center of visualization, which helps users observe a large visualization well.
- **Axis rotation** changes the position of an axis and thus changes the weight of a dimension or a set of dimensions to observe the correlation between dimensions.
- **Cluster marking or unmarking** mark the selected subset as a cluster or cancel the marked cluster.
- **Label loading** loads the clustering result produced by other systems.
- **Random rendering** changes all of the ζ parameters randomly in the same time and helps users find interesting visualization quickly.
- **Automated rendering** continuously changes one ζ parameter automatically. The user can switch to render the next dimension or the previous dimension, or jump to render a dimension by stopping current rendering, typing in the number of the dimension, and restarting the rendering. The dimension is numbered along the counter-clockwise direction. For example, in Figure 3, the dimensions s_i (I = 1..6) are numbered from 0 to 5 in the counter-clockwise direction.

ζ Adjustment

As we discussed, ζ adjustment is used to change the projection plane. ζ adjustment allows user to change the ζ values, which are represented by axis length(the scale of the absolute ζ value |ζ|), and axis color (green for positive values and yellow for negative values), thus increasing or decreasing the contribution of a particular attribute on the resultant visualization. In the extended version, we allow users to visualize the datasets that contain few categorical data attributes, by hashing the categorical data to a number in 0..n, where n is the number of all categories in the attribute. Then the numeric representation of categories is normalized to [0, 1] as same as the other numeric attributes. Groups classified by a particular attribute, regardless of numeric or numeric representation of categories, can be obviously observed when adjusting the corresponding. Especially, when continuously adjusting ζ of a categorical attribute that has a few categories, a user can see the groups identified by the categorical attribute moving in different directions and in different speeds. A visualization example of web log dataset provided by Jim Gray [] shown as in the Figure 7, the user scales up the “ClientIP” attribute from the initial layout to examine the distribution of the data points belonging to different ClientIP. With ζ adjustment only, we can see there are 2 groups.

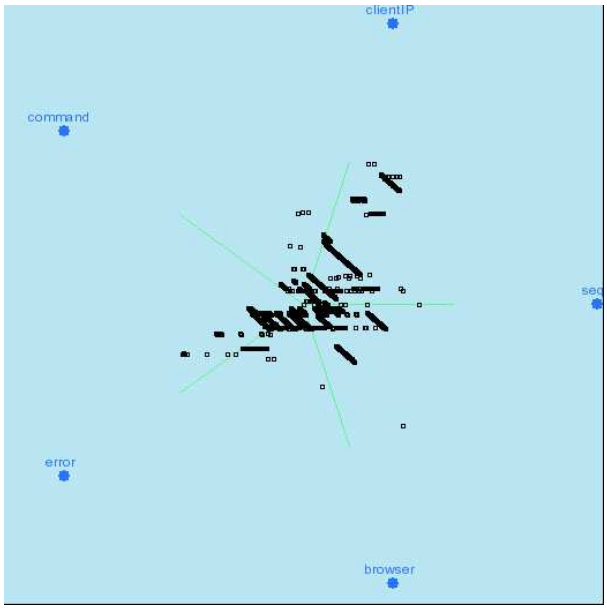

Figure 7a: The initial visualization of the web log dataset

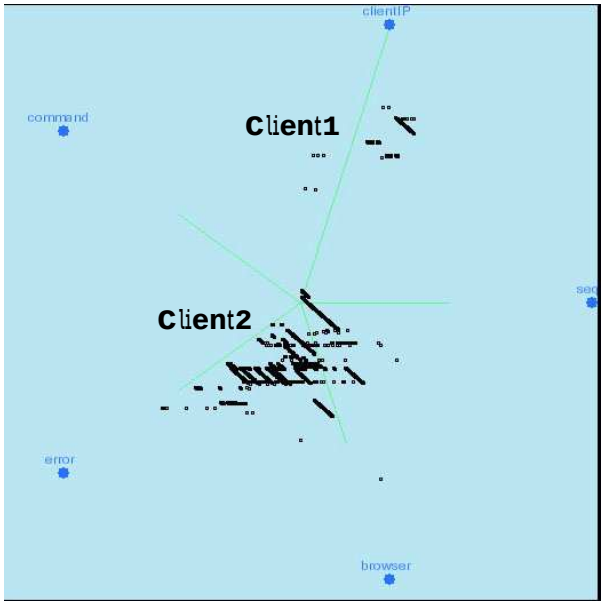

Figure 7b: Maximize the ζ value of ClientIP

ζ adjustment is also a natural way to interact with hierarchically expanding and collapsing clusters. In Figure 7, after scaling up the "ClientIP" attribute, a subsequent scaling of "Browser" attribute shows there are 3 sub-clusters (MSIE, Mozilla and Mozilla for NT) in the group Client2, but there is no more sub-cluster in the group Client1 (MSIE only).

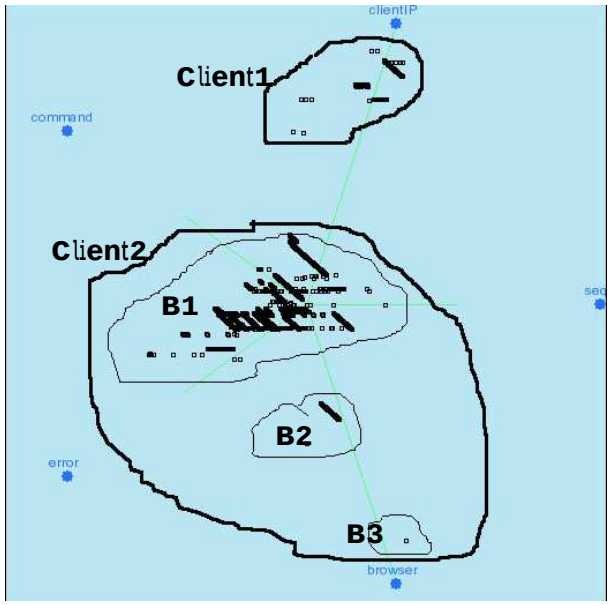

Figure 7c: Scaling up the "browser" attribute

To apply ζ adjustment, users operate on the ζ widgets, which are blue and at the end of each ζ axis. After loading a dataset, the axes (i = 1, 2, ⋯ , k), which correspond to the dimensions, are created, and the initial (i

= 1, 2, ‥ , *k*) (one per dimension) are set to 0.5. To change ζ values, set the operation mode to "change alpha" at first. By pressing and holding the mouse button on ζ widget, the corresponding ζ value is increased, and by pressing mouse button plus "Ctrl" key the ζ value will be decreased.

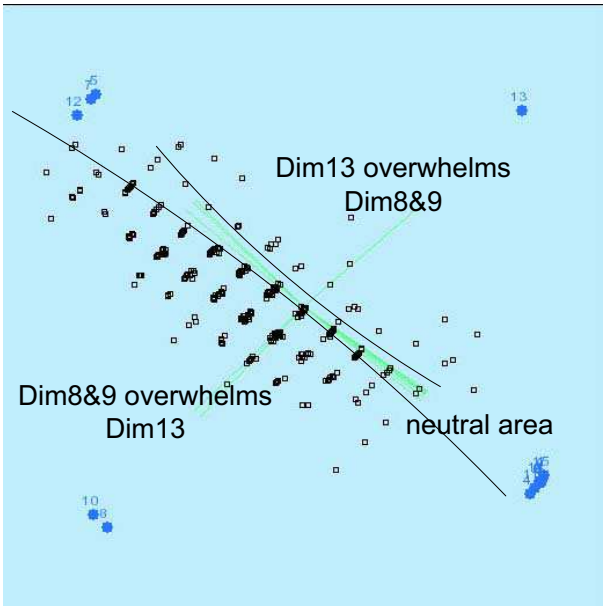

Figure 8: Axis rotation of the "voting" dataset

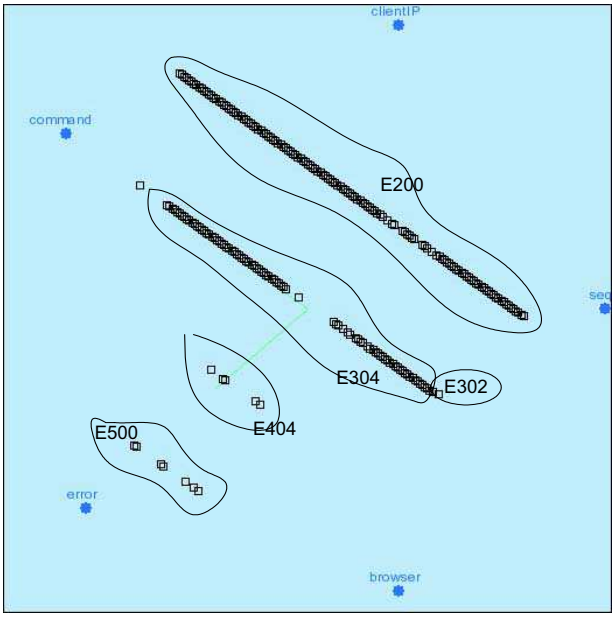

Figure 9: Axis rotation of the web log dataset

Axis Rotation

Axis rotation changes the direction of an axis, thus making a particular data attribute more or less correlated with other attributes. When several axes are rotated to the same direction, their contribution to the resultant visualization is aggregated. To observe the opposite effect of two sets of interesting attributes, for example, one is regarded as a positive set and the other as a negative set, the user can place the two sets in two reverse directions and deploy the other undesirable attributes in other directions. For an instance, in year-1984 house voting dataset we want to observe the tradeoff between the voting to the education expense (the attribute: education-spending) and the military/communication budgets (attributes: anti-satellite-test-ban and mx-missile). We rotate the axis (#13 education-spending) to the up-right corner and the other two (#8 anti-satellite-test-ban and #10 mx-missile) to the bottom-left corner. The visualization shows that the military/communication budgets overwhelm the education spending in most congressmen's consideration.

Axis rotation can also be used to observe the relationship between any two attributes in a precise way. The user can rotate the target axes to two perpendicular directions and then set the ζ values of other attributes to 0, which produces a standard 2D plot. For example, we want to observe the relationship between the attributes "Command" and "Error Code" in the above web log dataset, where the user can find which commands cause a certain error code, such as which commands causes the "500" error code, which implies that the web user tries to hack the web server system.

Subset Selection by Region

Vista system provides subset selection by free-drawing the enclosed region you want to observe. To begin subset selection, firstly the user needs to shift the operation mode to "subset selection". To draw the boundary, click the mouse on any beginning point on the boundary, then drag mouse and enclose the

boundary. The ending point does not need to be exactly on the beginning point. Subset selection will automatically enclose the area by connecting the ending point and the beginning point with a line segment. Subset selection can also be done on the selected subset, which enables the user to observe the dataset in a hierarchical way. Subset selection combining zooming is especially useful when exploring large amount of data points in a dense area. The example is the cluster rendering of dataset DS1-3D. The dataset has five clusters – one big spherical, two small spherical and two connected elliptical clusters. In the initial visualization (Figure 10a), we can see three clusters. The other two clusters may be hidden in the largest cluster. We select and zoom in the large cluster to see if there are other hidden clusters (Figure 10b). By using ζ parameter adjustment and zooming, together with subset selection, we can easily observe the detail of any

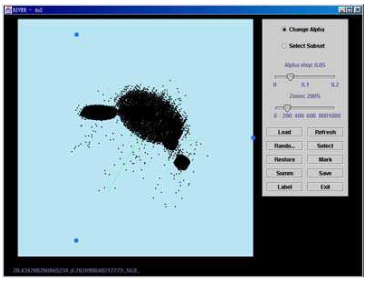

Fig10a. The initial visualization after loading the dataset

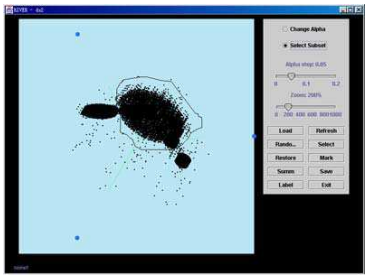

Fig10b. Select a subset

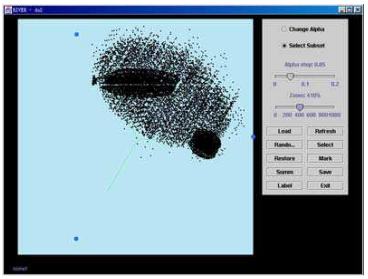

Fig10c. Show only the selected area and zoom in

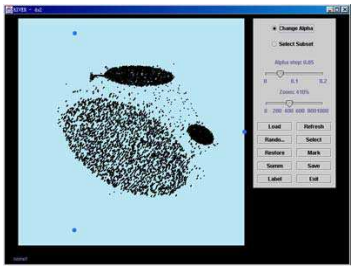

Fig10d. Adjust parameters

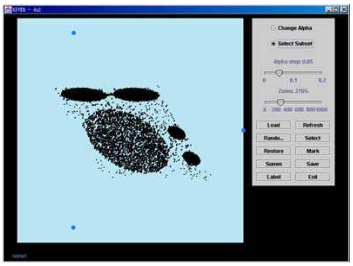

Fig 10e. Restore and zoom out, 5 clusters found

section. Figure 10c shows the detail of the select largest cluster in Figure 10b. In Figure 10c, we can see there are two overlapped clusters hidden in the large cluster. By adjusting the ζ values of different dimensions, we can move the overlapped clusters out of the large cluster. Finally, zooming out and restoring the shaded data points, we find the five clusters in visualization.

Subset Selection by Range

Range selection is a kind of subset selection, which selects a subset by specifying a range of one attribute. In current version, we only implement range selection of one attribute. In the future version, the Vista cluster rendering system can combine ranges of different attributes with AND or OR. To activate the range selection, click right mouse button on an alpha widget. The range selection dialog box will appear at the top-right corner of the screen. Because in the extended version, we allow to explore datasets that contains categorical data attributes, we provide two kinds of range selection dialog boxes for numerical and categorical data, respectively. The range selection dialog box for numerical data provides two sliders, allowing users to selection the lower and upper bounds of the range. The range selection dialog box for categorical data has a list of all categories in this attribute. The user can choose one or multiple categories. When the user changes the numerical range or category selections, the corresponding selected data points are shown in red color. The

user, then, can do all of the operations that require doing subset selection first. To recover the original unselected state, the user can open any range selection dialog box and press "Reset" button.

Cluster Marking and Unmarking
Marking a cluster requires selecting the subset first, which can be done by region selection or range selection. After selecting the subset, press button "Mark" in the button area to mark the cluster. To unmark a cluster, click mouse button on any point in the cluster when holding "Ctrl" key.

Automated Rendering
Automated rendering is mainly used in unsupervised cluster rendering, which automatically increases or decreases the alpha value of the target attribute – the value is increased to 1 first and then decreased to –1, repeatedly. Automated rendering starts at the dimension 0 by default, however, the user can type in any valid dimension into the dimension box to change the beginning dimension. Click the "Auto" button to start the rendering. The user can stop the current rendering by clicking the "Stop" button, or shift to render the next/previous dimension by clicking "Next" or "Prev" button. Automated rendering provides a convenient way to adjust alpha values.

Random Rendering
Random rendering automatically changes all of the alpha values in a random amount at the same time. After random rendering for some time, the user may find some interesting visualizations, which may not be found by step-by-step alpha adjustment.

Label Loading
Label loading can be used when the cluster label set exists. The cluster label set is abstracted from the dataset that has been labeled, or produced by other clustering systems. Label loading is usually used for supervised cluster rendering. After loading the label set, the data points in different clusters are shown in different colors and shapes. The user's task is to find the best boundaries that can separate all clusters well. The resultant visualization is saved as "cluster map" used for ClusterMap labeling algorithm.

Auxiliary Operations
- **Zooming**: The user drags the zooming slider and the visualization will be changed at the same time.
- **Focus Changing**: When some interesting parts move out of the visualization in the rendering process, the user can change the center of the visualization by click right mouse button on the new center while holding the "Shift" key.

5. *CLUSTERMAP* – A FAST AND FLEXIBLE DISK LABELLING ALGORITHM

In the clustering phase, visualization methods to find clusters tend to be slower than well-designed algorithms on small dataset, because human-computer interaction is obviously slower than automated computing. However, in 3-phase model for very large datasets, we argue that the most time-consuming part is disk labelling. The overall overhead of 3 phases is:

Sampling time + clustering time + disk labelling time

To explore potential performance improvement, we need to analyse the cost distribution of the three-phase model. Consider the dataset in the form of row by column, N is the number of rows in the entire dataset, and n is the number of rows sampled. Each row has k columns. We refer to a row in a dataset as a k-dimensional point in the data space. We first estimate the cost of each phase. At sampling phase, a reservoir-based random sampling costs $O(n(1+logN/n))$ [15], a typical automated clustering algorithm costs $O(n^2)$~ $O(n^3)$ and a reasonable labelling algorithm costs $O(N)$. We also need to investigate if the parallelism can be explored between the phases. Clustering algorithms usually require the global information about all points, however, when partitioning technique is applied to clustering algorithms, the clustering phase and the sampling phase can be partially overlapped. As the labelling phase needs to utilize the complete result of clustering, the clustering and labelling phases have to run in a serial order. Thus, when the size N of the entire dataset is large and growing, and the size n of the sample dataset is bounded by the maximum system capability, e.g. the limitation of memory, I/O and CPU, the cost of labelling phase will dominate the overall cost of the three-phase model. Figure 1 shows the difference between the three phases with some given (N, n) value pair, where $n<<N$.

Let $C_{three\text{-}phase}$ denote the overall cost of the three-phase model. Let C_s be the cost of sampling phase, C_c be the cost of clustering phase, and C_l be the cost of labelling phase. Let $a(n)$ denote the amount of cost saving when the sampling and clustering phases can be partially overlapped. The cost of the three-phase model can be represented by:

$$C_{three\text{-}phase}(N, n) = C_s(N, n) + C_c(n) + C_l(N) - a(n)$$

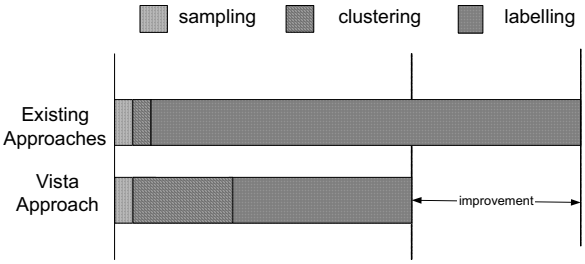

Figure 12: Expected improvement of the *Vista* approach compared to some existing approaches, given the value pair (N, n), $n<<N$

In the three-phase model, any effort to reduce the cost of sampling and clustering phases may easily be overridden by the potentially much higher cost incurred in the labelling phase. In contrast, any reduction on the cost of the labelling phase, namely, a smaller factor r of the cost (rN), may have a significant impact on to

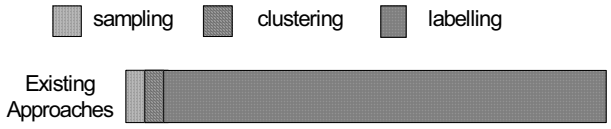

Figure 11: The time distribution of three-phase model with some value pair (N, n), $n<<N$

the performance of the entire process. With this cost analysis in mind, we expect the *Vista* approach improves the overall performance as shown in Figure 2. Concretely, the *Vista* clustering phase may cost more due to the additional latency introduced by visual interaction, and the *Vista* labelling phase will significantly reduce the

cost compared to existing labelling algorithms thanks to the *ClusterMap* labelling algorithm. As a result, the overall improvement made by the *Vista* approach can be significant for large N ($\varnothing 10^8$). We briefly review BIRCH and CURE's labelling phase here. BIRCH labelling is based on comparison to centroids of clusters. To label a point, it just compares the distances of this point to each centroid. The point is labelled with cluster id of the nearest centroid. CURE labelling can utilize the representative points produced by clustering. It looks up the nearest neighbour of this point among all representative points and then labels the point with cluster id of the nearest representative point. Because the total number of representative points is definitely larger than the number of clusters, CURE labelling costs more to label the whole dataset than centroid-based labelling. However, CURE labelling labels datasets more precisely, thus has lower error rate in general.

We propose the *ClusterMap* labelling algorithm, avoiding the tradeoff between the precision and performance. It runs faster and yields better results. The basic idea behind the *ClusterMap* is to promote the idea of "mapping is labelling". Briefly, with *Vista* system we can find a satisfactory mapping that discriminates clusters and outliers well, and the clustering result is encoded as a "cluster map" that includes the mapping parameters to create the final visualization and a 2D map that records cluster regions. The *ClusterMap* labelling algorithm utilizes the "cluster map" to label any item in the raw dataset. Concretely, it maps each data item in the raw dataset onto the "cluster map" and finds the corresponding label. Even though the map is created from the result of a sample set, it can describe the boundary of clusters precisely as long as the sample set preserves the cluster structure. The additional benefits are 1) the boundary can be adjusted and modified to adapt to any special situation, 2) and the outliers can be distinguished well.

The *ClusterMap* algorithm includes three steps: map construction from the *Vista* cluster rendering phase, map rebuilding, and map-based labelling. *Vista* cluster rendering and *ClusterMap* work together to complete the large-scale clustering process.

5.1 Map Construction

The process of constructing a cluster map is exactly the process of visual clustering with *Vista*. After continued visual exploration, the final step is to mark cluster regions. Although a user may see clusters as dense regions via visualization, it is not necessary to make the regions matching the boundaries exactly. You can extend the boundary slightly, combine two or more clusters together, or even split one cluster to several smaller clusters if necessary. After the cluster regions are marked, the entire display area can be saved as a width by height table (the map). Each grid of the table is labelled as the cluster ID if it is located on a cluster region, otherwise, labelled as outlier. For 800*600 resolution, the display area is about 520*520, and 688*688 for 1024*768 resolution. The area can also be scaled to be smaller by zooming in the visualization. To store such a "map" or rebuild it in memory does not cost too much at all.

All parameters used to create the final visualization are saved for map rebuilding. These parameters include:

- dimensions of visualization,
- resolution (zooming parameter),
- shift of the map center,
- maximum and minimum values of each numeric column for normalization,
- alpha values for B mapping,

5.2 Map Rebuilding and Labelling

Map rebuilding is a simple process that reads the cluster map and its parameters into memory, ready for mapping. Labelling process is the same as the mapping process in the *Vista* cluster rendering system. Each

data item from the raw dataset is processed by the modified max-min normalization and the *B* mapping, which yields a 2-d coordinate (x, y). Using the index of (x, y), we can find the cluster label in the cluster map, which is used to label the given data item. This process iterates over the entire raw dataset.

5.3 Complexity Analysis

We estimate the cost of *ClusterMap* (considering only map rebuilding and labelling) and compare *ClusterMap* with the existing labelling algorithms to see if *ClusterMap* is faster than the existing labelling algorithms.

Given a raw dataset, assume that, k is the dimensionality and N is the size of the raw dataset. We count the number of necessary multiplication to estimate the cost. For example, one k-d Euclidean distance calculation costs k multiplication. In the map rebuilding step, map rebuilding and parameter reading cost constant time, about 200ms for 800*600 resolution or 350 ms for 1024*768 (see experiment section). In the labelling step, for each data item in the raw dataset, the max-min normalization costs O(k) multiplication. The *B* mapping function costs O(k) multiplication respectively to calculate x and y coordinates. Searching for the cluster map to get the corresponding cluster ID for labelling costs O(1) time. Hence, the total cost for the entire dataset is O(3kN).

Let us use the same estimation to evaluate the cost of CURE (RPB) labelling and the centroid-based (CB) labelling. Assume that k is the dimensionality and N is the size of the raw dataset, c is the number of clusters of the raw dataset, and each cluster needs m representative points for description. To get the best performance of existing labelling algorithms, we use *kd*-tree [23, 24] to organize the representative points, the cost to find the nearest neighbour point in *kd*-tree is at least $log_2(cm)$ distance calculation. Thus the cost of CURE labelling is at least $log_2(cm)* kN$. As reported in the CURE paper, only when the number of representative points is great than 10 ($m>=10$), the CURE labelling can always get right clusters. Therefore, $log_2(cm)$ is usually great than 4, namely, the cost of CURE labelling is great than 4kN. Thus, *ClusterMap* outperforms CURE labelling in most cases, and at the same time it preserves the low error rate (see experimental results).

The cost of the centroid-based disk labelling is at least $log_2(c)* kN$, if c centroids are organized by *kd*-tree. However, when c is small, e.g., c<10, the direct search, which costs ckN, runs faster than the *kd*-tree search on average. Analytically, for large c, the CB labelling may run a little faster than the RPB labelling and the *ClusterMap* labelling. However, the CB labelling produces very poor results when the shape of cluster is non-spherical.

k	Dimensionality
N	Total rows in raw dataset
c	The number of clusters
m	The number of representative points in CURE
$f1$	The cost of *ClusterMap*, $3kN$
$f2$	The cost of representative labelling, $cmkN > f2 > log_2(cm)* kN$
$f3$	The cost of Centroid-based labelling, $ckN \geq f3 > log_2(c)*kN$

Table 1: Parameters and cost estimation of three labelling algorithms.

6. EXPERIMENTAL RESULTS

This section presents two sets of experiments. The main objective of these experiments is to evaluate the effectiveness and sensitivities of the *Vista* approach, in particular, to demonstrate that the *Vista* approach is significantly faster and yields lower error rate when the size *N* of the raw datasets is large enough.

The first set of experiments is to evaluate the usability of *Vista* system and to prove the effectiveness of *Vista* visual clustering on real datasets. The first prototype of the *Vista* cluster rendering system can deal with up to 50000 items and up to 40 dimensions in real time. In this range, the size of the dataset does not affect the performance of the clustering process nor the clustering result. For very large datasets, the raw dataset is first sampled to get a subset of representative samples, with less than 50000 items. Therefore, the categories of datasets, whether they are small or medium or large in size, will not make much difference in evaluating results. Therefore we choose several public-domain datasets that are popular in terms of the cluster irregularity and application-specific explanations, although they are small or medium in size. The second set of experiments is designed to evaluate the effectiveness of the *ClusterMap* labelling algorithm. We proceed in two steps. First, we discuss the experiments that compare the *ClusterMap* algorithm to the existing labelling algorithms, and demonstrate that the *ClusterMap* labelling usually has lower cost and error rate at the labelling phase. As we discussed earlier, the costs of the labelling algorithms are analytically linear, therefore, validating the effectiveness of the labelling algorithms using small/medium sample sizes is sufficient to understand and predict the common labelling behaviour on the large dataset. The second goal of the labelling experiments is to see how the error rate changes when the size of the raw datasets being labelled increases. Finally, we combine the two sets of experiments and show how much overall performance the *Vista* approach can improve when the datasets grow up to certain size.

6.1 Usability of *Vista* system – error rates of visual clustering phase

The *Vista* visual clustering system was implemented in Java. Our initial experiments were conducted on a number of well-known datasets that can be found in UCI machine learning database (www.ics.uci.edu). These datasets, although small in size, have irregular cluster distribution, which is an important factor for testing the usability of the *Vista* system.

It is well known that CURE can give better results than other existing algorithms by recognizing clusters in irregular shapes. To show how effective the *Vista* visual clustering is in terms of error rate reduction, we choose to compare the error rates of the *Vista* visual clustering with the CURE clustering algorithm. Two kinds of visual clustering methods are used in our experiments, one is unsupervised process and the other is supervised. In unsupervised visual clustering process, a user marks clusters based on the information obtained via dynamic and interactive exploration, such as dense area, point movement and statistical information. In this kind of exploration, visualization may trap into local minima that does not reflect actual cluster distribution very well, thus may lead to a higher error rate than the supervised clustering. In the supervised visual clustering process, the training datasets are labelled and the items in different clusters are visualized in different colors. The user's responsibility is to find a good visualization that distinguishes the labelled clusters in different regions. The labelled data may provide hints to direct the user to find a satisfactory visualization much quicker. Because of these advantages the supervised visual clustering usually gives better results than the unsupervised visual clustering. Both of the visual clustering methods can be used in practice to generate a "cluster map" for the labelling phase.

Table 2 shows the experiments on usability of *Vista* over six well-known datasets. Error rates of *Vista* are compared with error rates of CURE, where *N* is the number of rows in the given dataset, *k* is dimensionality

of the dataset, U.Su. denotes the unsupervised method, and Su denotes the supervised method. Interesting to note is that it takes a trained user 5 mins to 20 mins to find satisfactory visualizations for these datasets. The supervised clustering also costs less than the unsupervised clustering. We use 15 mins as the average of visual clustering cost, C_v. This set of experiments shows that *Vista* usually offers lower error rates than CURE on these real datasets.

Dataset	*N*	*k*	*Vista*(%)		CURE (%)
			U.Su	Su	
breast-cancer-wisc	699	10	16.7	4.3	36.6
crx	690	15	20.2	14.5	31.7
hepatitis	155	19	21.9	20.6	41.3
votes	435	16	25.5	5.5	38.2
iris	151	4	5.5	3.3	35.7
page-blocks	5473	10	13.0	7.0	53.4
heart	270	12	24.0	16.7	49.6
mushroom	8124	21	24.7	2.5	36.8
australian	690	14	15.4	14.4	35.7
Wine	178	12	7.9	6.2	34.3
Shuttle	14500	9	10.2	4.2	17.5

Table 2: Experiments on usability of *Vista* comparing error rates of *Vista* with CURE, *N* is the number of rows in dataset, *k* is dimensionality, U.Su. is the abbreviation of unsupervised, and Su is supervised.

6.2 Performance and error rates of labelling phase

We have discussed three different disk-labelling algorithms: Centroid-Based labelling (CB) in BIRCH [8], Representative-Point Based labelling (RPB) in CURE [3] and *ClusterMap*. In this section we study the performance and error rate of *ClusterMap* compared to the two other popular labelling algorithms: centroid-based labelling and CURE labelling.

CURE labelling
CURE clustering produces *m* representative points for each cluster. These representative points describe the convex of the cluster. To label a point, find the nearest neighbour among all representative points and then label the point with cluster ID of the nearest representative point.

Centroid-based labelling
Centroid-based labelling is simpler than CURE labelling. Rather than represent a cluster with *m* points, only the centroid point of a cluster is used to represent the cluster. To label a point, just compare the distance to each centroid. The point is labelled with cluster ID of the nearest centroid.

In the following subsections we first describe the datasets used in the experiments and the environmental setup for these experiments. Then we discuss the experimental results obtained over different datasets.

6.2.1 Datasets and Experiment Setup

Datasets:
Two datasets are used for the second set of the experiments reported in this paper. One is a simulated dataset that is derived from the dataset DS1 used in CURE. We refer to this dataset as DS1-5d in the rest of the paper. This dataset extends the original 2D DS1 to 5D to complicate the visualization and distance calculation. We

observed that all of the original distances in DS1 are greater than 1. To avoid changing the original cluster distribution, the values of the additional 3 dimensions are randomly selected in [0.01, 0.1], which do not change the basic distance relationship. The visualization of the result of applying the CURE clustering algorithm over the DS1-5d also shows that the clustering result is the same as that of DS1. DS1-5d has five regular clusters, including three spherical clusters, two connected elliptical clusters, and lots of outliers. The second dataset we use is a real dataset – Shuttle dataset (STATLOG version). It is a 9-dimensional dataset that has very irregular cluster distribution. There are seven clusters in this dataset, among which one is very large with approximately 80% of data items, and two are moderately large with approximately 15% and 5% of data items, respectively. The others are tiny. All the datasets have original labels, so we can perform error rate comparison on different labelling algorithms.

Experiment setup:
We run experiments on a PentiumIII700MHz PC. Operating system is Redhat7.1. All three labelling algorithms are implemented in C++. The CS Department of University of Wisconsin at Madison provided the implementation of CURE clustering. We run CURE clustering with the number of representative points set to 10, alpha set to 0.5 (shrink factor), and k set to the expected number of clusters. We use ANN (Approximate Nearest Neighbour) C++ library from University of Maryland at College Park to construct *kd*-tree for CURE labelling.

We choose to run the experiments on two datasets: DS1-5d and Shuttle. When we evaluate the experimental results of *Vista* and CURE, the constant parts – the cost of cluster map rebuilding and *kd*-tree building, are excluded from the graphs. However, we still list the numbers in discussion.

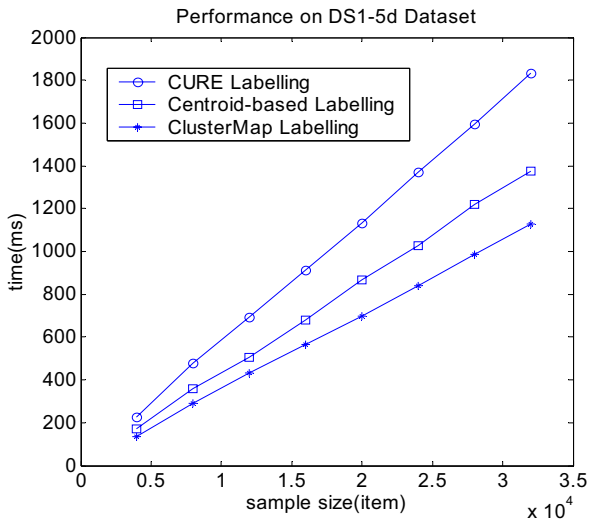

Figure 13: Cost on DS1-5d

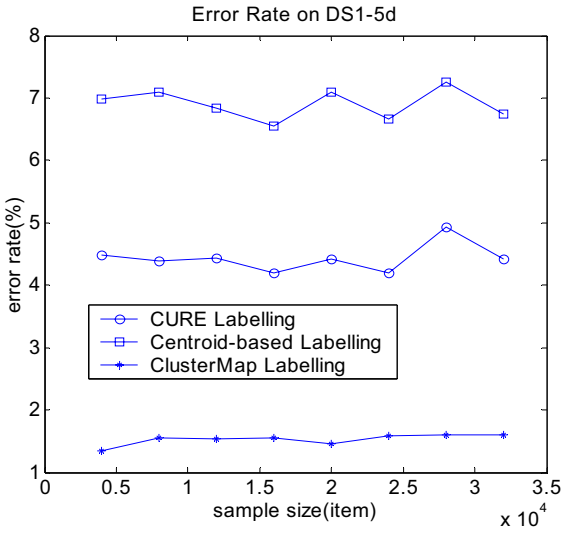

Figure 14: Error rate on DS1-5d

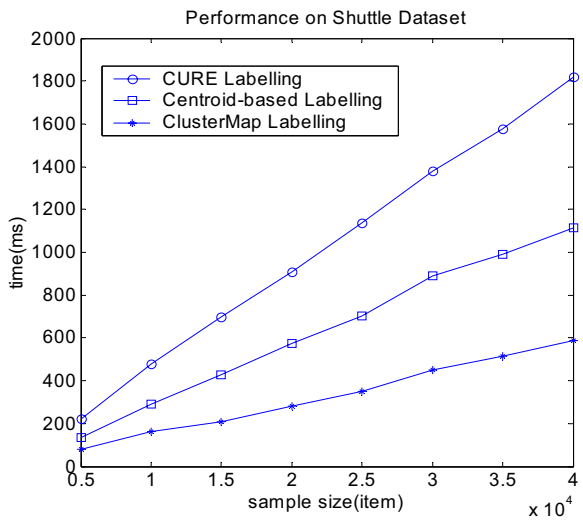

Figure 15: cost on Shuttle

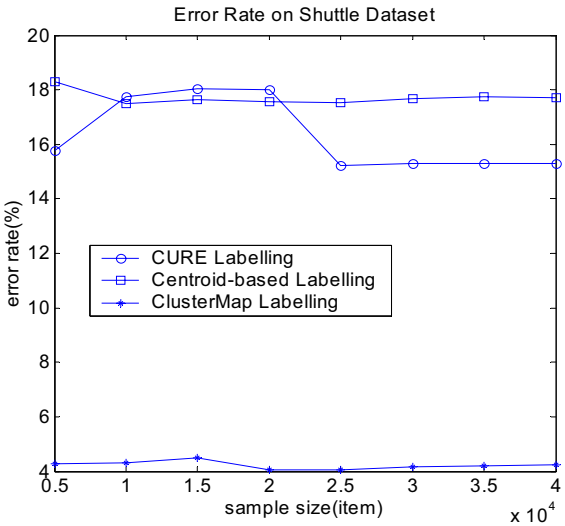

Figure16: Error rate on Shuttle

6.2.2 Performance Comparison of Three Labelling Algorithms on DS1-5d Dataset

We start *Vista* by loading the DS1-5d dataset. To find clusters of DS1-5d with *Vista*, we work with the original two dimensions of the dataset and set ζ values of the additional three dimensions to 0. We get the "cluster map" in the resolution of 688*688 (see Figure 17). The cost to rebuild the cluster map is about 340~360ms. In contrast the cost to build *kd*-tree is about 1~2ms. All clustering algorithms are performed on the dataset of 4000 –data items. The experimental result in Figure 13 shows that the costs of all three algorithms are linear to the size of the dataset. CURE labelling costs about 1.6 times more than *ClusterMap*. We also found that the Centroid-based labelling with *kd*-tree search costs more than with the direct search because of the small number of nodes in *kd*-tree. The Centroid-based labelling shown in Figure 13 uses the direct search, which still costs more than *ClusterMap*.

The DS1-5d dataset is also used to show the effect of outliers to the labelling algorithms. The error rate of CURE labelling is on average of 4.5% over DS1-5d; The error rate of the centroid-based labelling has the error rate of 6.8%; while the error rate of *ClusterMap* has only about 1.5% incorrect labelling. *ClusterMap* also shows a more stable error rate. The DS1-5d dataset has one large spherical cluster, two small spherical clusters, two ellipse clusters and lots of outliers. Centroid-based labelling suffers from the large circle cluster and outliers. Most CURE error labels are from outliers. Due to the space limitation, we only show the visualization of CURE labelling results. Visualization of CURE labelling on 8000-item subset (Figure 17) shows outliers around each cluster are labelled as this cluster.

5.2.3 Experimental Results on Shuttle Dataset

Shuttle dataset has very irregular clusters (Figure18). We run *ClusterMap* on a resolution of 520*520. The cost to build the map is about 190~200ms. The cost to build *kd*-tree is about 1~2ms. All clustering algorithms are performed on a 5000-item subset. The result also shows the costs of all three algorithms are linear to the

size of tbe Shuttle dataset (Figure 15). But this time CURE labelling costs 2.6 times as much as *ClusterMap* and the Centroid-based labelling uses the direct search again because there are only 3 centroids.

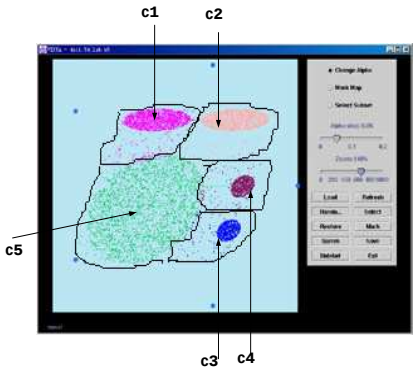

Figure 17: Visualization of labelling result on DS1-5d

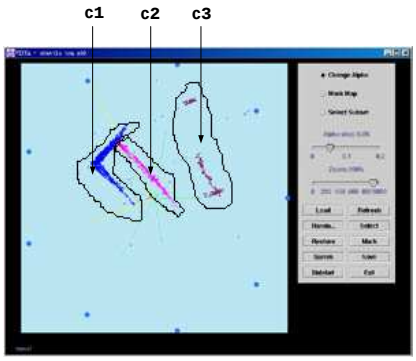

Figure18: Visualization of correct labelling result on Shuttle

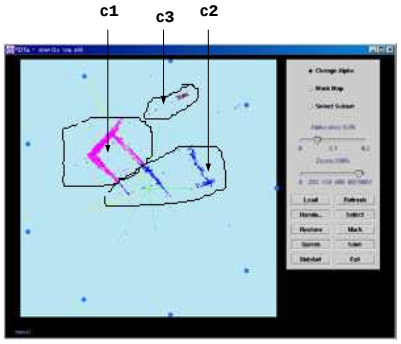

Figure19: Visualization of CURE labelling result on Shuttle

We use the shuttle dataset to evaluate how the error rate from the clustering phase affects the labelling algorithms. Figure 16 shows that the error rate of CURE labelling is on average of 17% over the Shuttle dataset and the centroid-based labelling has an error rate of 18%. *ClusterMap* has only 4.2% of incorrect labelling, much lower than the other two algorithms. The high error rate with the CURE labelling is primarily caused by the irregular shapes of clusters, leading to wrong convex description for labelling.

Let us take a closer look at the three large clusters shown in Figure18. Cluster c3 actually has two parts; c1 and c2 are connected; c1 is in triangle shaped distribution. CURE clustering simply cannot give correct clustering result. In the visualization of CURE labelling on 10000-item subset (figure19), we can see CURE divides the original cluster c3 into two parts and cannot discriminate c1 and c2. We give the correct centroids for centroid-based clustering, however, centroids cannot represent the irregular cluster distribution well, which turns out the high error rate in labelling result.

6.2.4 Overall performance of the Three-phase Clustering

Recall Section 5, we have derived the cost equations for the existing automated approach and the *Vista* approach. If we only consider the clustering and labelling phases, we have the equations:

1) $C_{existing} = C_c + C_l$, $C_l = r_1 * N$
2) $C_{Vista} = C_v + C_b + C_m$, $C_m = r_2 * N$

$C_{existing}$ is the general cost of existing automated approaches. C_c is the cost of clustering phase that usually utilizes an automated clustering algorithm. C_l is the cost of the labelling phase in O (N), which can be represented as $r_1 * N$, N is the number of rows in entire dataset, and r_1 is the factor determined by the cost of translating the items from strings to numbers and the distance calculations, both are to some extent affected by the dimensionality of the dataset.

C_{Vista} is the total cost of the *Vista* approach in clustering and labelling phases. C_v is the cost of visual clustering. C_b is the cost of rebuilding cluster map and C_m is the cost of *ClusterMap* labelling. Similarly, we use r_2*N to represent C_m and r_2 is determined by the cost of translating the items from strings to numbers and the mapping parameters. In other words, the dimensionality of the dataset and the boundaries for normalization are two important factors for calculating r_2.

Due to the space restriction, in this paper we only include the performance comparison of the *Vista* approach with the CURE approach on the shuttle dataset. CURE clustering on the shuttle dataset takes about 10sec with the following parameters: the sample size n = 5000, the number of representative points = 10, partitions = 5 and $\zeta = 0.5$.

As mentioned earlier, the average cost of visual clustering, C_v, is about 15min, or 900 sec. The cost of map rebuilding, C_b, is 200ms-350ms, which can be ignored compared to the large value of C_v. From the performance results shown in Figure 15, we can deduce the parameters r_1 ° 0.000046 and r_2 ° 0.000015 for the shuttle dataset if the cost is calculated in unit of seconds. Therefore, we have following cost equations for an extended dataset "shuttle", which has a larger number of rows, e.g. $>10^8$:

$C_{CURE} = C_c + r_1*N$ ° 10 + 0.000046*N (sec)
$C_{Vista} = C_v + C_b + r_2*N$ ° 900 + 0.000015*N (sec)

Because human interaction is much slower than automated algorithms, for small or median size datasets, the CURE approach is certainly faster than *Vista* approach. However, when the dataset size grows up to $3*10^7$ rows, the *Vista* approach begins to outperform CURE approach (Figure 20). The *Vista* approach will save significant amount of time when the dataset size becomes larger. For example, when the size grows to $2*10^8$, the *Vista* approach saves about 5300 sec, which represents a saving of more than a half of the CURE cost. Similarly, for DS1-5d dataset, we have r1 ° 0.000057 and r2 ° 0.000035. With $2*10^8$ items, the *Vista* approach saves nearly 3500 sec for DS1-5d (Figure 21).

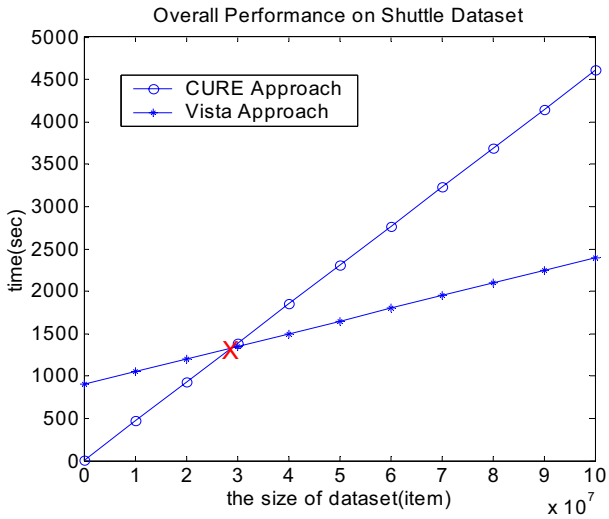

Figure20: Overall performance on shuttle

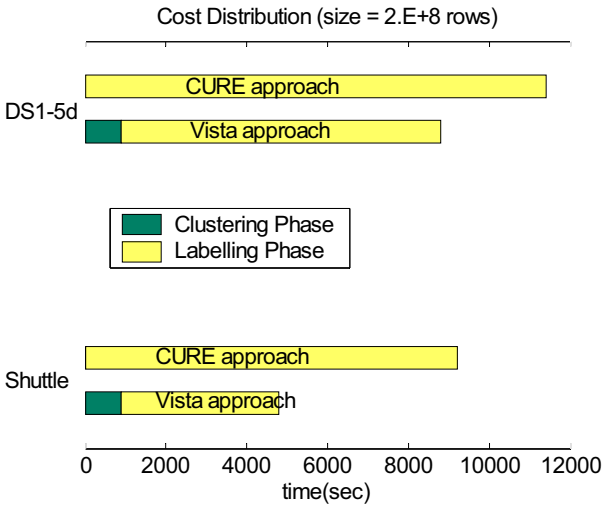

Figure 21: Cost distribution (the cost of automated clustering is too little to be seen)

6. Related work

Data Clustering has been extensively studied over the past decade. However, most of the past research has been focused on efficiency of clustering datasets of small or medium size. Issues on large-scale clustering have only been explored recently. CURE [3] and BIRCH[8] are the two representative clustering algorithms aiming at the large-scale clustering problem. Both of them utilize the three-phase model with different optimisation emphasises. CURE uses random sampling to get a representative subset, which is then used as the input of a hierarchical clustering algorithm. To speed up the clustering phase, CURE also takes advantage of partitioning technique. However, there is no effort in CURE [3] to speed up the disk labelling process. As a result, even though CURE offers a fastest algorithm in the clustering phase, compared with *Vista*, CURE's overall performance for large scale clustering is not optimised. BIRCH's contribution focuses on the sampling phase. It presents a new technique – summarization (or aggregation), which organizes sub-cluster information in CF (Cluster Feature) tree, to reduce the input set for clustering algorithm. Similar to CURE, a hierarchical clustering algorithm is applied to the CF tree to get the clustering result. Again, BIRCH did not pay attention to the disk-labelling phase.

Random sampling techniques have been studied for long time, among which reservoir based random sampling is regarded as an efficient approach. J.Vitter [15] proved that the lower bound of the reservoir based sampling is O($n(1+logN/n)$), where N is the number of items in the given dataset and n is the size of its sample subset. He also proposed an algorithm - Z, which can reach the lower bound. As far as the problem of preserving cluster structure after sampling is concerned, the authors of CURE presented a theorem to describe the relationship between the sample size, the raw dataset size and the minimum size of clusters.

Various efforts are made to visualize multidimensional datasets. The early research on general plot based data visualization is Grand Tour and Projection Pursuit [12]. Since there are numerous projections from a multidimensional data space to a 2D space, the purpose of the Grand Tour and the Project Pursuit is to guide user to find interesting projections. L.Yang [13] utilizes the Grand Tour technique to show projections of datasets in an animation. They project the dimensions to coordinates in a 3D space that is similar to the mapping used in our system. However, when the 3D space is shown on a 2D screen, some axes may be overlapped by other axes, which makes it hard to perform direct interactions on dimensions. Moreover, the "clusters" seen in Grand Tour are not proved as real clusters. Star Coordinate [14] is a visualization system designed to visualize clusters. The mapping functions and some of the interaction techniques are to some extent similar to the *Vista* approach. However, the emphasis and contributions in [14] are solely on visualization design. There is no discussion on how the visualization results will be used for large-scale clustering. Other techniques, such as Scatterplot matrices, coplots, and prosection[2] create static visualization only, thus they do not provide enough cluster information by visualization.

Because the existing large-scale approaches tend to ignore the labelling phase, only a handful number of labelling algorithms are reported up to date. To our knowledge, the two popular disk-labelling algorithms are the centroid-based labelling and the representative-point-based labelling. Both of them are distance-comparison-based, which turn out to be costly and introduce more errors into labelling result. Our experiments show that the *ClusterMap* labelling is effective and fast and plays a critical role in speeding up the visualization-based clustering systems.

7. Discussion

We have presented the *Vista* approach and described the two important and complimentary features of the

Vista approach, namely reducing the error rate through the *Vista* visual clustering, and using a fast disk-labelling algorithm 4 *ClusterMap* to label the entire dataset, thus improving the overall performance of the three-phase clustering model.In this section, we discuss a number of frequently asked questions, some of which are our ongoing effort and others are parts of our future work.

Can the *Vista* Approach always be efficient and effective?
A short answer to this question is no. Most traditional clustering algorithms assume that clusters of the datasets have spherical shape and similar size with very few outliers. Such regularity can be approximately represented by some probability distribution functions. The automated algorithms together with the centroid-based labelling are more efficient, compared to the *Vista* approach, when the datasets in practice have or are assumed to have such cluster distribution, i.e., there are no irregular clusters in the datasets. However, many real-world datasets often observe non-spherical clusters, some even in irregular shape. Most of the automated algorithms generate higher error rates in clustering results even though they provide fast algorithms at the clustering phase. The *Vista* approach aims at providing more flexible, and yet reliable and efficient ways to explore the datasets of irregular clusters. Its clustering rendering system provides richer interactive operations to help users understand the datasets better, and thus leads to better clustering results, whereas its *ClusterMap* labelling algorithm outperforms all existing disk labelling algorithms, and the performance gain from the labelling phase is significant when the raw dataset size N is much larger than the sample data size n. Even for small or medium datasets, where the advantage of the *ClusterMap* labelling algorithm may not be significant, the *Vista* approach may still be beneficial for error rate reduction if the applications emphasize on high precision output of the clustering results for those datasets that contain irregular clusters.

A user can observe the cluster distribution by adjusting the parameters to see the dynamic and continuous changes in visualizations. However, a linear model does not ensure that all cluster distributions can be projected to one plane, where the clusters are well separated. Sometimes, the user knows the skeleton of the cluster distribution by interactions but he/she cannot find a very precise visualization. In this case, the user has to adopt an approximately satisfactory visualization instead. This approximation in practice is still very effective to yield low error rates for most datasets. To explore the very precise boundaries of the clusters for the very complicated cluster distributions, two or more planes may be needed and an extended *ClusterMap* algorithm needs to be designed accordingly.

Will the *Vista* visual cluster rendering system have acceptable performance?
A short answer to this question is yes. People often concern about the efficiency of visual exploration of large datasets, especially by utilizing the existing interaction techniques. Concretely, take the *Vista* system as an example, it seems hard to find a satisfactory projection plane by adjusting ζ parameters. We have done some experiments, in which an average user, trained in 1 hour or so, can find the satisfactory visualization for most experimental datasets in an acceptable time period, i.e., no longer than 20 minutes. The *Vista* cluster rendering system has another interesting feature. That is, the cost of visual clustering does not vary much in terms of the size of the sampled dataset. When the cluster distribution is clear, users can find the satisfactory visualization very quickly regardless the size of the processed dataset. In contrast, most of the automated algorithms on a dataset of near 50,000 items may cost more time to find satisfactory clusters than the visual clustering. Currently we are working on designing more effective interaction techniques, which is easy to use and can help users find results more quickly. These techniques will be incorporated into the next generation of the *Vista* system.

How does the *Vista* visual rendering system handle high dimensional datasets?

The first prototype of the *Vista* system can visualize datasets of up to 50 dimensions. For the datasets that have higher than 50 dimensions, one way to address the problem is to reduce the dimensionality by applying a distance-preserving transformation, such as FastMap [11], before applying the k-dimension to 2-dimension mapping in *Vista*. Such a transformation can also be used to transform non-Euclidean distance space to Euclidean distance space. However, this solution has a number of problems due to the dimensionality reduction transformation. First, it may introduce additional errors into the clustering phase. Second, it makes it less intuitive or sometime harder to interpret or understand the meaning of the interactive operations. Third, it may increase the cost of the *ClusterMap* labelling algorithm when labelling the entire dataset of dimensionality higher than 50. It is widely recognized in the information visualization community that visualizing very high dimensional datasets is an important open problem.

Although the idea to visualize datasets and find clusters in visualization is not new, a system like *Vista* cluster rendering was not published before. We will try to release it as a platform for further research on the visual clustering issues. We hope our efforts will boost the research on this promising area.

8. Conclusion

Most of the existing three-phase clustering approaches have been solely dedicated to performance enhancement at the clustering phase. Very few have addressed the problem of how to reduce the overall cost of large-scale clustering. We have described the *Vista* approach for efficient and flexible clustering of large datasets. Our approach is built upon the three phase clustering model with two unique characteristics. First, we promote visual clustering instead of automated algorithms at the clustering phase with the objective of reducing the error rate of the intermediate results. Second, we develop a fast disk-labelling algorithm that encodes the visualization result in a "cluster map" and utilize the map to label the entire dataset. We have shown two important results from our initial experiments on a number of real-world data sets. . First, the *Vista* visual clustering framework , either supervised or unsupervised exploration, can improve the precision of the intermediate clustering results. Second, the disk-labelling algorithm *ClusterMap* outperforms the existing disk-labelling algorithms on any size of datasets in most cases. A unique feature of the *Vista* approach is its ability of combining visual cluster rendering with a fast disk-labelling algorithm to improve the overall performance of the three-phase clustering model for large datasets, while maintaining the lower error rates. The first prototype of *Vista* is available at disl.cc.gatech.edu/ *VISTA*.

Our research on *Vista* continues along two directions. On the theoretical side, we are interested in investigating issues related to improving the speed of visual cluster rendering while maintaining the low error rate. On the practical side, we are interested in exploring the capabilities of *Vista* on large-scale web log datasets and intrusion detection -systems.

Acknowlegement
The first author would like to thank his Master advisor, Prof. Ashim Garg of SUNY at Buffalo. Working with Prof. Garg on the interactive visual framework inspired him to explore the use of visualization on large-scale clustering. We also appreciate Prof. Vipin Kumar who provides us the implementation of CURE. This research is also partially supported by a NSF CCR grant, a NSF ITR grant, and a DoE grant.

REFERENCE:

[1] D.Bhadra and A.Garg. “An Interactive Visual Framework for Detecting Clusters of a Multidimensional dataset”, Tech Report2001-3-10, CSE Department, SUNY at Buffalo, 2001
[2] W.S.Cleveland. “Visualizing Data”, AT&T Bell Laboratories, Hobart Press, NJ.1993.
[3] G.Guha, R.Rastogi, and K.Shim. “CURE: An efficient clustering algorithm for large databases”, in Proc. of the 1998 ACM SIGMOD
[4] S.Guha, R.Rastogi and K.Shim. “ROCK: A robust clustering algorithm for categorical attributes”, in IEEE Conference on Data Engineering, 1999
[5] A. Inselberg and B.Dimsdale. “Parallel coordinates: A tool for visualizing multi-dimensional geometry”, in Proc. Of IEEEVisualization’90, pp361-375, 1990
[6] L.Kaufman and P.Rouseeuw. “Finding Groups in Data: an Introduction to Cluster Analysis”, John Wiley&sons, 1990
[7] M. Halkidi, M. Vazirgiannis, I. Batistakis. “Quality scheme assessment in the clustering process”, in Proc. of PKDD, Lyon, France, 2000
[8] J.Larkin and H.A.Simon. “Why a diagram is (sometimes) worth ten thousand words”, Cognitive Science, 11(1): 65-99, 1987
[9] R.Ng and J.Han. “Efficient and effective method for spatial data mining”, In VLDB94, pages 144-155, 1994
[10]G.Sheikholeslami, S.Chatterjee, and A.Zhang. “Wavecluster: A multi-resolution clustering approach for very large spatial databases”, In Proc. VLDB98’, 1998
[11] T, Zhang, R.Ramakrishnan, and M.Livny. “BIRCH: An efficient data clustering method for very large databases”, In Proc. Of SIGMOD96’ pages 103-114
[12] I. S. Dhillon, D. S. Modha& W. S. Spangler. “Visualizing Class Structure of Multidimensional Data”, In Proc. of the 30th Symposium on the Interface: Computing Science and Statistics, May 1998.
[13] D. Keim. “Visual Exploration of Large Data Sets”, Communications of the ACM. August 2001/ V. 44. No. 8
[14] C. Faloutsos, K. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Tradditional and Multimedia Datasets," in Proceedings of 1995 ACM SIGMOD, vol.24, no.2, p 163-174.
[15] J. Wang, X. Wang, K. Lin, D. Shasha, B. Shapiro, K. Zhang. “Evaluating a Class of Distance-Mapping Algorithms for Data Mining and Clustering”, Proceedings of the fifth ACM SIGKDD international conference on KDD, 1999
[16] Ester, M., Kriegel, H-P., Sander, J. and Xu, X. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, in Proc. of the 2nd International Conference on KDD, Aug, 1996.
[17] Cook, D.R, Buja, A., Cabrea, J., and Hurley, H. “Grand tour and projection pursuit”, Journal of Computational and Graphical Statistics, V23, pp. 225-250
[18] Charu C. Aggarwal, Philip S. Yu. “Finding Generalized projected clusters in high dimensional spaces”, in Proc. of 2000 ACM SIGMOD, 2000
[19] Jeffrey S. Vitter. “Random Sampling with a Reservoir”, ACM Transactions on Mathematical Software, Vol. 11, No.1, March 1985, Pages 37-57
[20] Liu, H. and Motoda, H. “Feature Extraction, Construction and Selection: A Data Mining Perspective”, Kluwer Academic Publishers, Boston, 1998.
[21] Joseph B. Kruskal and Myron Wish. “Multidimensional scaling”, SAGE publications, Beverly Hills,1978
[22] A. Jain and R.Dubes. “Algorithms for Clustering Data”, Prentice hall, Englewood Cliffs, NJ, 1988
[23]